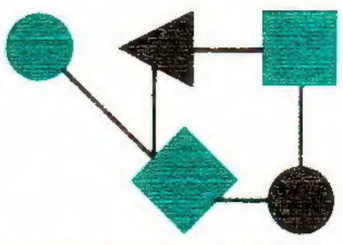
CONNEXIONS



The Interoperability Report

November 1987 Volume 1, No. 7

ConneXions - The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, 21370 Vai Avenue, Cupertino, CA 95014, USA 408-996-2042.

ISSN 0894-5926

From the Editor

This month we bring you an article describing the work being done by the Network Management Working Group. The article is by one of the Working Group's members, Amatzia Ben-Artzi of Sytek, Inc. He is one of the major developers of Sytek's TCP/IP product line, and has previously taken part in the NetBIOS effort leading to RFC 1001 and RFC 1002.

Greg Minshall concludes his description of *tn3270*. The second article is entitled "IBM 3270 Data Stream over Telnet".

In our series "Improving your TCP", Craig Partridge looks at ICMP Source Quench and how it can be used to improve performance and avoid network congestion.

RFC 1025 entitled "TCP and IP Bake Off" was recently issued. This RFC puts a bit of history of the early work on TCP/IP into the RFC record. The Bake Offs were testing sessions for the early implementations, and this document lists the test suggested for those session. Some of these test may be of interest for new implementations. The tests are also interesting in light of the current work on developing testing and certification laboratories.

We are pleased to announce that effective immediately, the annual subscription rate for *ConneXions* will be reduced to $100/year (plus $50 additional/year for international addresses). This will allow us to broaden the readership and extend vital information to others who need it. Current subscribers have already received a rebate coupon which may be applied towards additional subscriptions or registration at December TCP/IP conference.

We are only one month away from the *2nd TCP/IP Interoperability Conferenc*e, which will be held at the Hyatt Regency Crystal City, Arlington VA, from December 1-4, 1987. The program has now been finalized and on page 15 we list some of the key points.

If you haven't registered to attend the conference yet, you should do so immediately as space is limited. Contact us for more information at 408-996-2042.

See you in Crystal City!

The Network Management Working Group

by Amatzia Ben-Artzi, Sytek, Inc.

Some time ago, RFCs 1001 and 1002--describing NetBIOS over TCP/IP were released. This was the first time that vendors--not researchers--joined together and published RFCs that were mutually accepted by all vendors ("all" meaning those who took part in the process).

Encouraged by this success, an agreement was reached to proceed to work together on an RFC for Network Management. From the very beginning it was felt that the NetBIOS effort (that took almost a year to complete) was nothing but a warm-up compared to the effort involved in Network Management. The main reason being, that while the NetBIOS effort was a technical challenge, but of a well-defined scope, "Network Management" is an unbounded term. Above and beyond the technical challenge, it involves conceptualization of the operational models, architecture of global communications networks and even operating culture. A technical solution cannot be developed without first putting it into some agreed-upon context.

What should be developed?

Not without sweat and tears, the group reached the (unavoidable) conclusion that what we are looking at is not *an* RFC, but a *set of* RFCs. The set reflects a conceptual model for operation, that is briefly discussed in the sections below. The set of RFCs identified to be developed includes:

- Overview, Architecture and Services
- Management Protocols
- Structure of the Management Information
- Software Distribution and Downloading

All together, for the initial phase, the above set includes 12 (twelve!) different RFCs.

Development Direction

The magic word "ISO" is very appealing to many vendors as well as users--there is a wide spread belief that "one day we are going to conduct all our communications activities using ISO standards". The problem here is that this "one day" is *not bound in time*. Another desire shared by many (at least those who do not suffer from the NIH syndrome) is *not to re-invent the wheel* and take advantage of whatever work is already available. The Gateway Monitoring Group (Craig Partridge of BBN and Glenn Trewitt of Stanford) with their HEMS (High-Level Entity Management System, See *ConneXions* Volume 1, No. 2) appeared to be an excellent starting point for development, only that they did not develop their system with ISO in mind...

In the TCP/IP community, we believe in "getting things out of the door quickly", therefore "waiting" for ISO makes no sense. On the other hand we do not want to re-do our entire work when this day does arrive. So what *should* we do? The approach seems to be the following: Define the architecture and the model for operation in a way that is as close as possible to ISO (as we understand it), and within this model work our way for an effective, short-term solution. We would like to have a *migration path* to ISO.

The model

Network Management information is viewed as a property of the communication layers. In addition there is another special "Layer" which is the "system". Each layer contains a separate (logical) entity called a Layer Management Entity (LME) that is in charge of the management activities *relevant to this layer*. The consolidation and coordination of the different layers' information is performed in each system by a process (again, logical) called the System Management Application Process (SMAP). Systems are tied together by a Manager-Agent relations, where one side is viewed as the "Master" and the other one as the "Slave". While there is nothing *inherent* in any system for a node to be only a Manager or only an Agent, and these relations can be dynamically changed or re-assigned by the system, it is typically expected that "end-nodes"--such as computers, terminal-servers, bridges and gateways--will play the "agents", while specialized management centers (such as a Sun Workstation) will play the "managers".

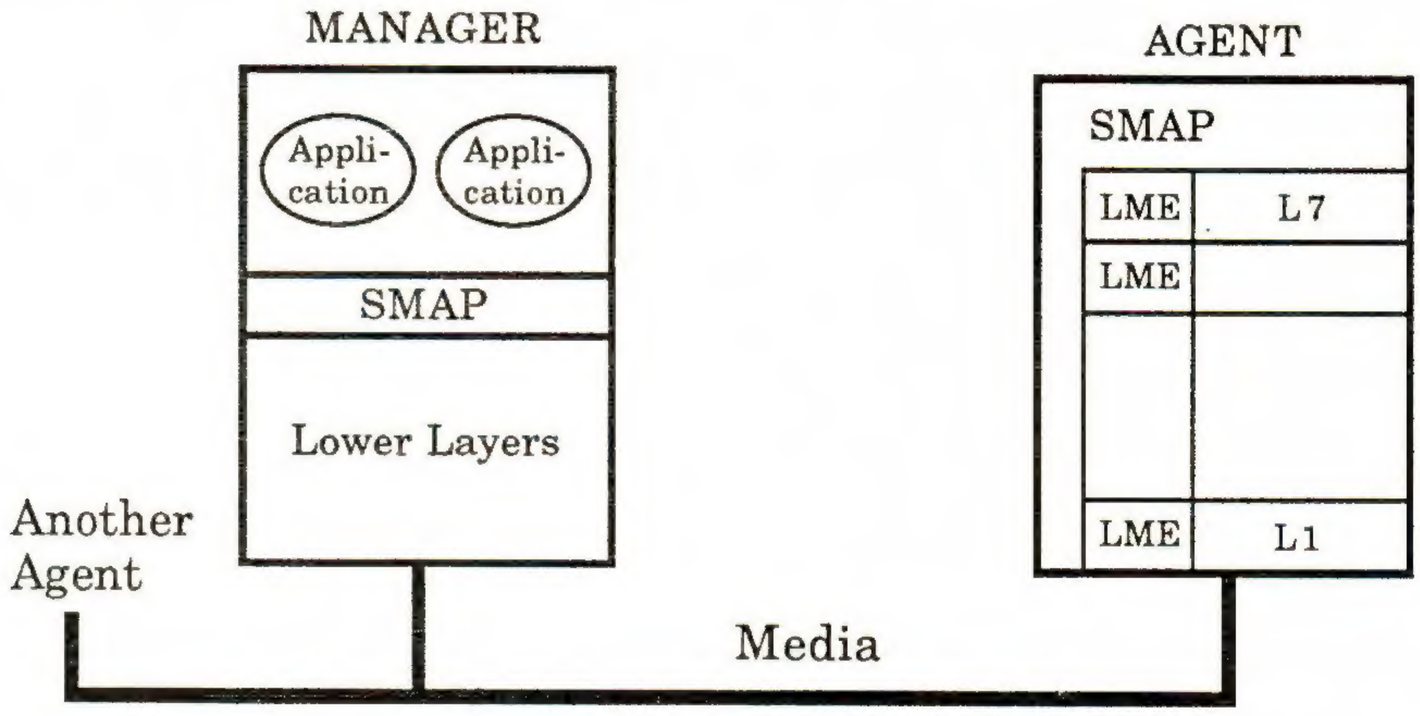


Figure 1: Network Management Model

For those of us who are involved in management of large-scale systems, something may be missing in the above model--at least if we understand it in a simplistic fashion. Today's networks require the ability to divide the system into different *domains of management*. Such domains may be buildings, departments or what have you. At the same time we may very well still want a centralized point of control (or "Focal Point") where *global* management information can be obtained and actions can be taken. This is shown in Figure 2 on the next page. (The machines labelled "M" represent passive "Monitor Agents" and the ones labelled "MGR" are Managers).

Building a system

How does this system fit into the model portrayed above? Recall that there is nothing in the model which says a node must be either a Manager or an Agent. This means that a Manager may have a number of nodes act as "agents in its domain", and at the same time this Manager can "report" to a higher-level Manager, thereby taking the role of an Agent. Figure 3 illustrates this structure for a dual-domain system.

continued on next page

Network Management Working Group *(continued)*

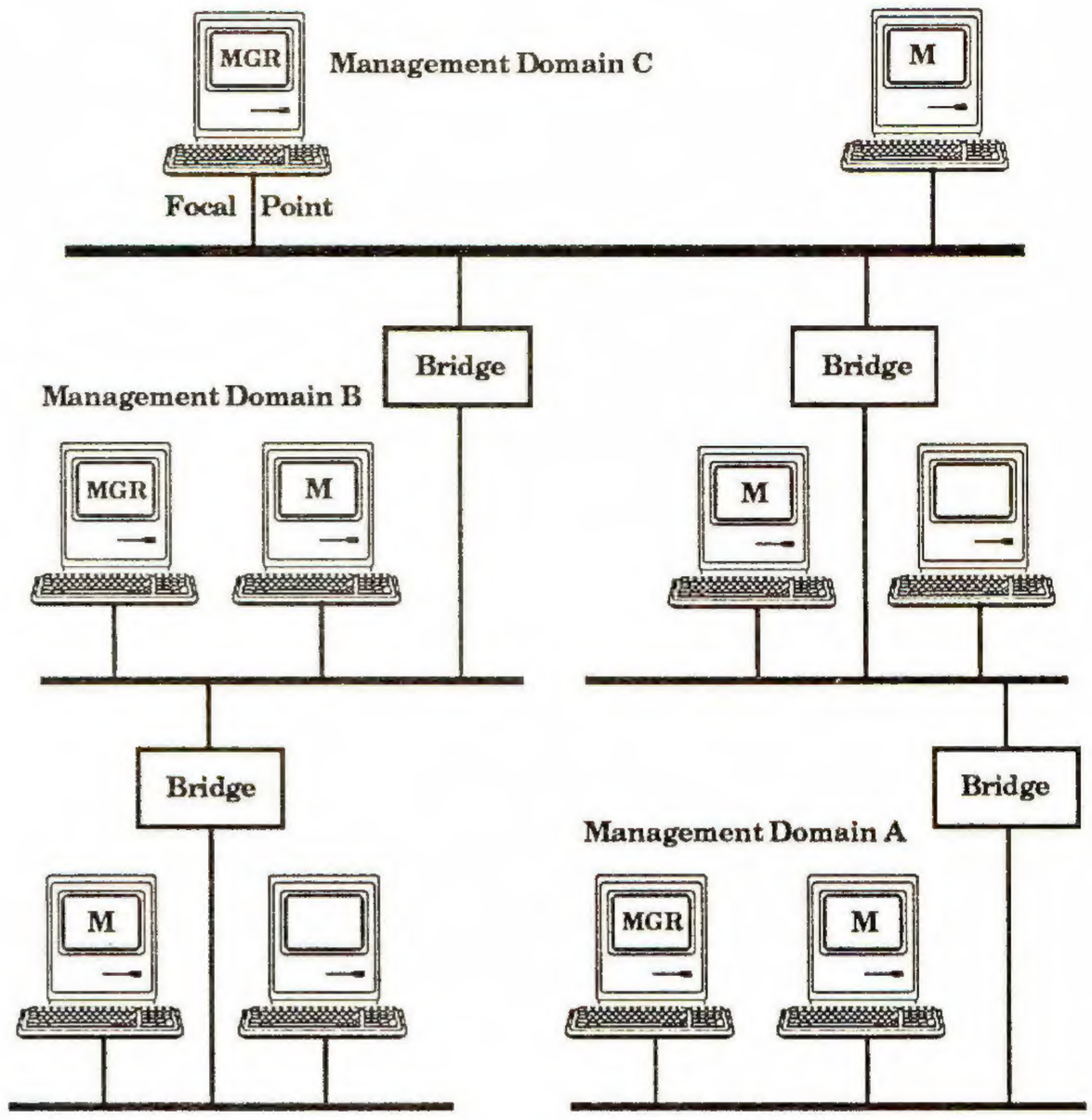


Figure 2: Management Organization

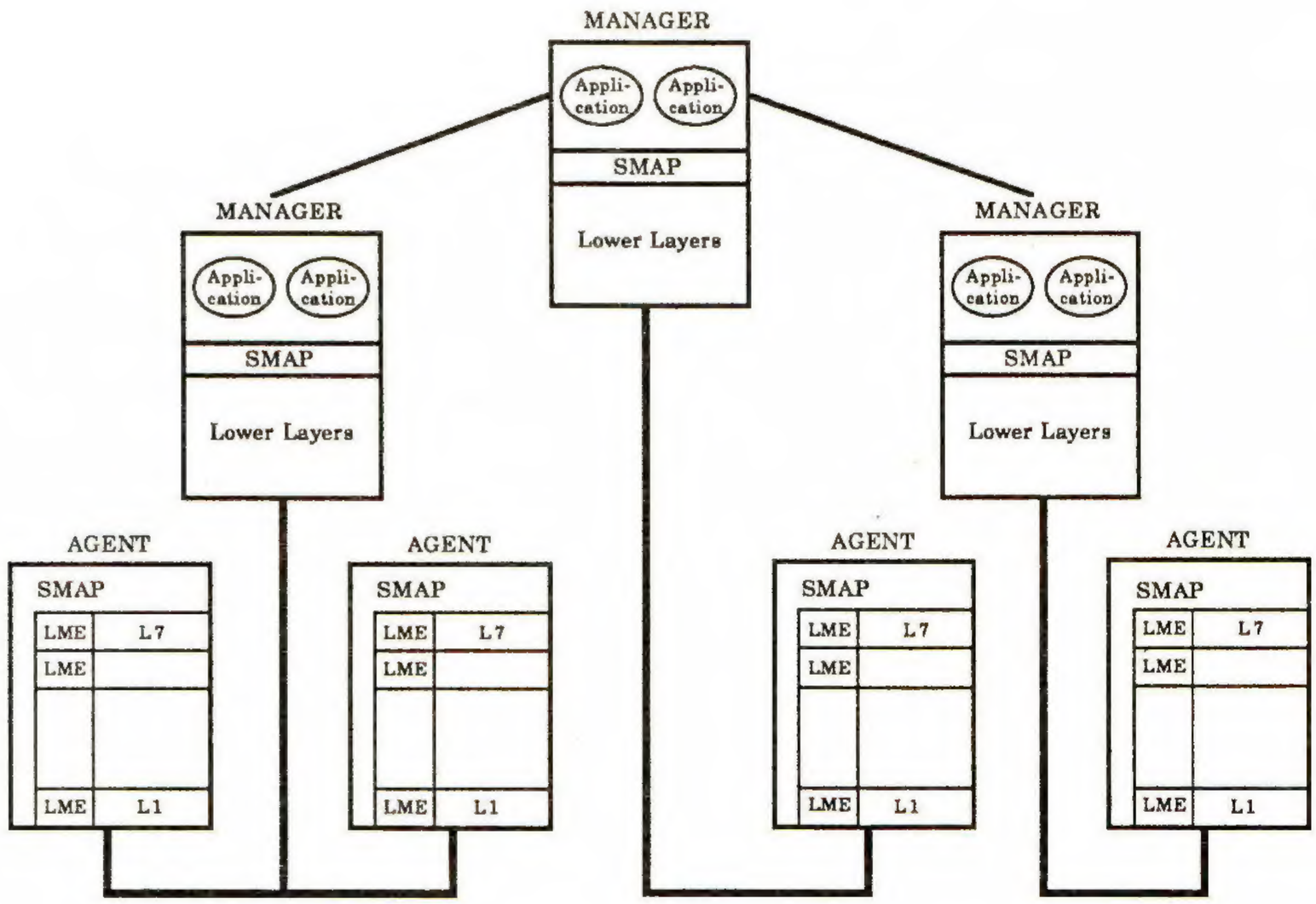


Figure 3: Management Hierarchy

Communication between systems

The communication (i.e. exchange of instruction, queries and information) between the systems occurs between the SMAPs. For example, if a Manager needs information about some parameters at the Network Layer, its SMAP sends an appropriate message to its peer SMAP (the one that resides in the end-node of interest). This SMAP "consults" with the Network Layer's LME, and the appropriate response (if any) is sent back from the end-node's SMAP to the Manager's SMAP and then delivered to the application.

To make this operation feasible in a multi-vendor environment, several aspects need to be considered:

1. Applications should be able to invoke a specific service in a standard way, that will be well understood by the "other" side.

2. The protocol that "carries" the service primitives should be common to both sides.

3. The reference to a specific piece of information should unambiguously identify exactly the same information in all systems.

Service Interface

The Service Interface is one of the most important things to standardize. It allows an application to issue functions that are recognizable by the other side. Such "functions" include GET info, SET info, perform ACTION, report EVENT, etc. The services used follow closely those defined by ISO in their "Common Management Information Services" (CMIS--ISO DP/9595). A draft RFC that defines the common services is currently being circulated.

Protocol

ISO is working on a management protocol called "Common Management Information Protocol" (CMIP--ISO DP/9596). It uses a fully established system for its underlying communication (i.e. all seven layers). For partial systems, or intermediate systems, that may work in a connection-less mode, the term "thin stack" is commonly used and refers to all seven layers in a "shrunken" form. However, this work is far from being complete, therefore there is an attempt to adopt the HEMP/QUERY of the HEMS system to be used as the management protocol between SMAPs.

SMI

SMI is a famous (!?) acronym for Structure of Management Information. What is really needed is a standard way through which a manager can address a very specific object in some end-node. The behavior of this object should be well understood (e.g. counters that roll-over, or tide-marks that "stick" at the top). All systems should have the same view of such an object and return information in the same way.

Tree structure

Management Objects, within each system, are organized in a tree structure. The top branches from the root may possibly distinguish DoD/TCP suite from say DoD/ISO objects. As we traverse the tree down, we may see layers as branches, and within each layer, a subsequent, layer-defined sub-tree. Examples may be configuration parameters (further broken down into actual parameters), routing tables (broken down into columns and fields) and so on. Individual objects are placed at the leaves of the tree. Reference to an object is made through specifying the entire path from the root to the leaf--uniquely identifying the object. Operations should be allowed on some intermediate, non-leaf, nodes on the tree. In this case the objects addressed are all those that are leaves to that sub-tree. (Creating a "wild-card" effect). There are also on-going discussions on specifying a "short-cut" path, where the starting point is not the absolute root, but some intermediate, agreed-upon node.

Both in the ISO arena, and in the TCP/IP arena, the SMI issue is not yet well developed and significant amount of work is still to be done.

tn3270 Part Two: IBM 3270 Data Stream over Telnet

by Greg Minshall, UC Berkeley

Last month we looked at the IBM 3270 architecture and introduced the concept of a Network Virtual Terminal (NVT) in the Telnet protocol. This time we will further examine the implementation of 3270 operation over Telnet.

DET Option

In the case of the IBM 3270, the Native Mode of operation clearly involves leaving NVT mode. However, after leaving NVT mode, the data stream which flows over the connection looks so different from Network ASCII that there has been some controversy about the desirability of sending a 3270 data stream Telnet. Within the extended definition of the Telnet protocol, there is a capability known as the data entry terminal option, DET, which attempts to provide a new model terminal which might be a better match for terminals in the same class as the IBM 3270.

The advantage which would come from using the DET model is that, in theory, all such data entry terminals would be served by one standard protocol, and thus all such terminals would be able to interoperate with any server host supporting DET mode.

There appear, however, to have been two major problems which have kept people from making extensive use of the DET model in the past. First, the DET model is complicated, and thus non-trivial to implement. Second, the NVT model is the happy meeting ground for many terminals and systems, and for this reason, one is willing to give up some (or in some cases, much) of the functionality of one's specific terminal for the benefit of generality. However, the DET model does not (currently) allow one access to very many systems, and so one is not overly anxious to transform from 3270 to DET at one end (thus losing some 3270 function), in order to transform from DET to 3270 at the other end.

The assumption of this article is that, for whatever reason (be it marketing, technical, or pragmatic), allowing an IBM 3270 to communicate in Native Mode is as good an idea as allowing any other terminal to communicate in Native Mode.

So, having determined that there is a need for a client Telnet program, (which is servicing a user with an IBM 3270), and a server Telnet program, (which is running on a host willing to talk in 3270 mode), to agree to communicate in 3270 Native Mode, we are faced with two questions: 1) How will the two ends agree to enter 3270 Native Mode?, and 2) What *is* "3270 Native Mode"?

Note that if one end of the connection enters native 3270 mode while the other blithely remains in NVT mode, no useful communication will occur and, most likely, the end user will be highly perplexed as to the type of failure occurring.

Initiating the dialog

In order to enter 3270 Native Mode, there are two pre-conditions. First, the client must have informed the server (using the Telnet terminal type option) that the Telnet user is using an IBM 3270 terminal. There is a specific list of terminal types which correspond to IBM 3270 terminals of various types and models.

Telnet End Of Record command

The second precondition is that the client and the server must have both agreed to use the Telnet end of record (EOR) command (technically, this means that each end agrees to assign some arbitrary meaning to Telnet EOR commands received, and to send Telnet EOR commands at some arbitrarily decided times; we will see below that in the case of 3270, these "arbitrary" times are not at all arbitrary).

Having established these two preconditions, 3270 Native Mode is entered when both sides agree to enter Telnet binary mode. (3270 Native Mode is exited when either side leaves Telnet binary mode).

What 3270 Native Mode consists of is not as intuitive as one might think. For most ASCII terminals the data stream can be defined as that appearing at the RS232 connector on the back. No such single data stream exists for 3270 terminals. Depending on how the 3270 is connected to the host, the data stream may look (slightly) different: locally attached 3270's see one data stream, BSC 3270's see another, and SNA 3270's see yet another.

Defining 3270 Native Mode

However, the data stream which the programmer sees is fairly consistent across all 3270's. This data stream, in the outbound direction, consists of the operation code, WCC, etc. In the inbound direction, the data stream consists of an AID, followed by specific data defining the (recently modified) contents of the screen. This data stream, then, is taken to be what is meant by "3270 Native Mode".

Once 3270 mode has been entered, the outbound 3270 data stream consists of an opcode, followed (in the case of a write command) by a WCC and optional write data. The particular segment of data stream is delimited by Telnet EOR commands. Thus, after entering 3270 mode, the first data (i.e. non-Telnet command or parameter) byte sent from server to client is presumed to be a 3270 opcode. The next 3270 opcode will be that byte which immediately succeeds the next Telnet EOR command found in the data stream. The inbound 3270 data stream is similarly delimited by Telnet EOR commands. After entry into 3270 mode, the first data byte sent from client to server is presumed to be an AID byte which, depending on the specific AID key, may be followed by a cursor position report and 3270 data.

Development at UCLA and Wisconsin

These protocols and concepts were developed jointly by researchers and developers at the University of Wisconsin at Madison and at the University of California at Los Angeles. The Wisconsin group was developing, under contract to IBM, a TCP/IP package for IBM's VM/CMS operating system (this package, known at one time as Wiscnet, has been recently enhanced and release by IBM as a Program Offering). The group in Los Angeles had, again under contract with IBM, developed a TCP/IP package which now runs under IBM's MVS operating system (this package, while not widely available directly from IBM, has been remarketed by various vendors).

continued on next page

IBM 3270 Data Stream over Telnet *(continued)*

Current protocol issues in 3270 mode

As of this writing there are many implementations of both server and client Telnet programs which are willing to negotiate 3270 mode. These versions mostly interoperate. That the two initial versions (those developed at Wisconsin and at UCLA) were basically compatible is, in large part, responsible for this interoperability. In addition, the widespread use of the Wisconsin product (and IBM-supplied derivatives) on many university campuses and the availability of 3270 emulators (like *tn3270*; see below) has prompted many suppliers to follow the protocol specification outlined in the last section. However, there are still some problems with and/or questions about the protocol.

Lack of standard specifications

The first, and most glaring, problem is that the protocol is not formally documented. There is no single source (such as an RFC) which defines the protocol. This is a problem for any vendor contemplating providing 3270 service over Telnet; they may well feel they are trying to conform to a moving target. This is also a problem for end-users trying to specify 3270 service as a requirement in an RPQ; the contract issuer would like to, say, require compatibility with RFCxxxx. The lack of such an RFC hampers these purchasing efforts.

Robustness issues

A second problem is that entering 3270 mode, while apparently robust in the current environment, is not robust in the face of client and/or server software changes. Above we noted that one side entering 3270 mode while the other stays in NVT mode was likely to be confusing to the terminal user. In theory, both sides have to agree to use the Telnet EOR option before entering 3270 mode; however, since some early implementations did not require the EOR option to be explicitly negotiated, there are still implementations (including the *tn3270* implementation described below) which will flip into 3270 mode without EOR being negotiated. Additionally, the EOR and binary modes are general Telnet functions; it is easy to conceive of a server willing (and, even eager) to negotiate either or both modes. The current approach, which does not involve any 3270-specific enhancements to the base Telnet protocol, is pleasing in its use of general facilities; whether this aesthetic can be maintained is to be seen.

Terminal types

Currently, a server interrogates terminal type information from a client by asking to see the next terminal type in a list which the client is willing to advertise. A client might thus report three or four terminal types; when there are no more terminal types to report, the client replies "unknown" to all further requests to send more terminal types.

There are some conditions under which a server, having requested to see the entire list may wish that it had agreed to use the first, second, whichever, terminal type offered by the client. With the current Telnet terminal type protocol, this is not possible. One possible enhancement to the general Telnet protocol, with possible benefit to both the 3270 world and the non-3270 world, would be some means of allowing the server, having seen the whole list, to suggest a terminal type for the client to imitate.

Quering for terminal type

Additionally, the current list of 3270 terminals (as defined in the Assigned Numbers RFC) is deficient. The deficiency, however, is not in the number of terminal types listed, but rather in presuming that any list can describe each possible type of terminal which will support IBM 3270 protocols. For example, with many of the newer IBM 3270 terminals, the graphics capability of the screen can be queried by the host application server "on the fly", and many host applications are coded to support a wide range of screen geometries. A current proposal is to allow the server (or host software behind the server) to query the terminal to find out its operating characteristics. Technically, this is done by sending a read partition-query or read partition-query list structured field to the terminal. The terminal, in response, returns a structured field containing a query reply to the server.

Using queries and query replies, a server can determine all operational characteristics of the terminal. The catch here is that only certain, newer, IBM 3270 terminals have the capability to deal with structured fields; older terminals will have something akin to cardiac arrest if presented with a structured field.

One current proposal here (by Michael A. Stein of UCLA) is to define (and document) exactly one additional terminal type for 3270 mode: ibm3270q (say). This would imply an IBM 3270 terminal which could deal with structured fields at least to the extent of replying to a query (the reply to the query can contain enough negative information to prevent any other structured field types from being sent to the terminal).

Error reporting

One other problem relates to the ability of a client, or the client's attached terminal, to report error information back to the server (for possible forwarding on to the application program driving the 3270). When 3270's are connected in a traditional manner, there is a way for a terminal to say "stop, I don't like what I was told to do, and here's why". With the current protocol, there is no such ability. The most the client can do on an error condition is relay some hopefully useful message to the user and then shut down the connection. It is very possible that this is all that is needed, but there needs to be some checking to be sure.

Printers

Finally, in a traditional IBM environment, each terminal (or, more to this point, printer) has a unique name by which it is known within its installation (or the network on which it operates). So, when a printer is powered up, some process (having previously acquired the right to serve as the driver for this printer) is informed "printer XX now ready for printing".

However, when a printer is (somehow) connected via Telnet to a host it is assigned some random name. There is currently not much activity in connecting printers via Telnet to IBM hosts, but if interest is generated, then the question of how the printer is to indicate, to the Telnet server, what function it is to accomplish, and what process to notify of its availability, will come up.

continued on next page

IBM 3270 Data Stream over Telnet *(continued)*

Past work at UC Berkeley

At UC Berkeley, IBM mainframes (running VM/CMS and MVS) have been in use for many years. The main installed terminal base, however, has been ASCII terminals, especially within the academic community. Actual IBM 3270 terminals have only been used by computer center staff and by some administrative data processing users.

Yale IUP

In order to support ASCII terminals, we acquired a product developed at Yale University known as the Yale IUP. The Yale IUP runs on an IBM Series/1 computer and performs a mapping between ASCII terminals and the 3270 data stream. The product allowed an installation to tailor mapping of key sequences received from the terminal into 3270 keys, and to define sequences which, for each terminal type, allow for fullscreen painting of the screen. In addition, the Yale IUP, supported various extensions to the 3270 data stream which were deemed appropriate in the case of an ASCII terminal emulating a 3270.

Wiscnet

After having used Series/1's with the Yale IUP to perform protocol conversion for several years, the University began to become aware of the need to provide networking functions between our various computers. After some discussion, the University decided to standardize on the TCP/IP protocols as the means of ensuring interoperability amongst the various computers. At about this time, IBM "granted" the University a large CPU on which to run VM/CMS. In searching for an existing TCP/IP for CMS, we discovered a development project at the University of Wisconsin which had as its goal the production of TCP/IP for CMS. After some negotiation between the universities of California and Wisconsin and also with IBM, we acquired an early version of the Wisconsin code known as Wiscnet.

Initially, it seemed that our problems were over. However, after actually installing and beginning to use Wiscnet, we became aware of the difficulties of using line-oriented Telnet to talk with CMS. Almost all CMS applications are designed to operate with a 3270 terminal, and they do not deal well with line-oriented operation (especially when this line-oriented operation is disguised, as in the Wiscnet case, as full-screen operation).

tn3270

At this time (in the early fall of 1984), the University began a development project to develop a program (now known as *tn3270*) which would run under 4.2/4.3 BSD UNIX and would, in conjunction with the Wiscnet software, emulate a 3270 terminal over a Telnet/TCP/IP/Ethernet connection between a UNIX workstation and an IBM host. One of the design goals of *tn3270* was that it look like the Yale IUP (at least those features of the Yale IUP which were in common use at the Berkeley).

Additionally, we hoped to overcome one of the main problems we had experienced with the Yale IUP: an individual user's inability to have his/her own keyboard mapping file.

Around this same time, there was similar development work in progress at the University of Wisconsin (by Marvin Solomon) and at Rice University (by Paul Milazzo).

The end result of our development work was a program which has been distributed by the University for the past two years. The program is distributed with Berkeley 4.3 BSD UNIX, and is additionally available separately from the University in a public service distribution.

The original *tn3270* emulated an IBM 3277. The most recent production version of *tn3270* emulates an IBM 3278 (models 2, 3, 4, or 5; these correspond to various screen sizes). Additionally, *tn3270* implements some of the Yale extensions.

Future work at Berkeley

As of the summer of 1987, a beta-test version of *tn3270* has been developed and distributed by the University. This version has several key features. First off, it will run on an IBM PC as well as on UNIX workstations. Next, it incorporates an Application Programming Interface, API, which is intended to be compatible with the IBM API (note, however, that we have not had the resources to test our interface against the IBM interface).

Finally, one particular Yale enhancement, transparent mode, has been brought into conformance with the Yale IUP. Transparent mode allows CMS programs to pass raw ASCII data through to the terminal. Many applications, including graphics programs, make use of this feature, and the previous inability of *tn3270* to support this mode correctly was somewhat of a thorn in our side.

The beta-test version is now completed. It is not clear at this time whether this will be brought up to production release level (which includes new documentation, etc.), though it will be put into production at Berkeley (and other sites, undoubtedly).

Standing in the way of a new production release are two other development projects we would like to undertake.

X-windows

The first project is to produce a version of *tn3270* which works in a Native Mode with the X windowing system from MIT's Project Athena. The X windowing system (aka X) is a device independent, networked windowing system which allows for text and graphics to be displayed on high resolution graphics screens. The advantages of a native X version of *tn3270* are that updating the screen should be much faster than in a standard UNIX version of *tn3270* (due to some inefficiencies of the UNIX terminal interface which X avoids), and that a native X version would allow us to provide a 3179G graphics capability within *tn3270*.

The 3179G, although now replaced by a newer model, provides for bitmap and vector graphics; X is well suited to this style of graphics interface. (In the PC world, Microsoft Windows would be the rough equivalent of X; it is unlikely that we will do any development for this until after Version 2.0 of Microsoft Windows has been released).

SRPI

The second development is to offer a SRPI interface via *tn3270* (in both the PC and Unix environments). Even though the beta-test *tn3270* supports an API, experience has shown that programming applications using API is very labor-intensive. SRPI provides a better model for programming cooperative PC/mainframe processing (and hence for programming cooperative UNIX/mainframe processing) tasks.

continued on next page

IBM 3270 Data Stream over Telnet *(continued)*

SRPI is essential in the long range. However, progress in implementing SRPI is currently hampered by IBM's inability to disclose the 3270 data stream flows which carry SRPI requests over the 3270 connection. Knowledge of these flows would allow us to program a SRPI interface in conjunction with *tn3270*, and yet continue to use the IBM provided CMS (or TSO, for that matter) routers and servers. For higher level services on a PC, the goal would be to use the software (now) provided by IBM as part of the CMS (or TSO) servers. On a UNIX workstation, brand new requesters [and servers also, as the IBM server/requester protocols are proprietary (and likely to remain so)] would have to be written.

To summarize our development plans, we will likely provide a native X version of *tn3270*. Shortly after this, we hope to provide 3179G functionality within the X version. Finally, we will continue to press IBM to disclose those portions of the SRPI interface needed to implement SRPI inside *tn3270*.

Obtaining *tn3270*

The source for *tn3270* is available by either of two paths:

1. Via Arpanet, by anonymous FTP to host ucbarpa.berkeley.edu
   Retrieve (in binary mode) pub/tn3270.tar.Z or pub/tn3270.tar.

2. From the Campus Software Office
   UC Berkeley, California 94720

Enclose a check, payable to "Regents of the University of California", for $100.00 (US). Specify "*tn3270*" on your order form. Orders are normally filled on an AT-style (1.2MB) diskette; as an option, a tape may be requested. This second form will result in the purchaser receiving bug fixes and notices of new versions of the program automatically for some period of time. *tn3270* may be copied and redistributed. The program will compile and run on most BSD UNIX systems. The program will also compile and run on a PC (or compatible) which:

a) has the MicroSoft C compiler version 4.0,
b) has the MicroSoft MASM 4.0,
c) has the Ungermann-Bass smart TCP/IP Ethernet board, and
d) has the Ungermann-Bass C runtime library (MSC 4.0)

GREG MINSHALL is a programmer at the University of California at Berkeley.

Improving Your TCP: Handling Source Quench

by Craig Partridge, BBN Laboratories Inc.

The topic of *network congestion* and *congestion control* has become very important to many users of networks. The reason is simple: many IP networks are suffering from their own popularity and often encounter periods where the demand exceeds the available bandwidth. In such situations the network and its users must take steps to reduce the demand, or the network can enter a state called "congestion collapse", a condition where the network is so flooded with data that almost no data gets through. Congestion collapse is a stable state -- a network will stay in this dire condition until some action is taken to reduce the demand.

ICMP Source Quench

In the TCP/IP protocol suite, the approved weapon against congestion is the Internet Control Message Protocol (ICMP) Source Quench Message. When a gateway or host determines that it is receiving data faster than it can handle, it may send a Source Quench message to the host which is sending the data. When a host receives a Source Quench it is expected to take actions to reduce the amount of traffic it sends to the host or gateway which generated the Source Quench. Unfortunately, many TCP implementations ignore the Source Quench message. In the rest of this article, we look at various approaches for handling a Source Quench, with frequent suggestions for further reading.

The place to start reading on congestion control is John Nagle's RFCs: RFC 896 and RFC 970. RFC 970 is a proof that congestion is always a problem in packet switched networks. RFC 896 is a detailed discussion of how TCP/IP networks can become congested and suggests ways TCP should react to congestion. Nagle observed two different causes of congestion: tinygrams and lack of support for Source Quench.

Tinygrams

Tinygrams, datagrams containing very little data, encourage congestion because they flood the network with inefficient packets (packets in which the control information in the headers dwarfs the information being sent). Tinygrams are typically generated by character-at-a-time applications such as Telnet. Nagle suggests a method for reducing tinygrams by changing TCPs to refuse to send new data until previous data has been acknowledged.

Changing the window size

The second problem Nagle identified was TCP implementations which did not adapt to congestion when they received a Source Quench. He proposed a simple scheme which is still the baseline solution -- the solution everyone accepts as reasonable, if not necessarily optimal. In his scheme, whenever a TCP connection discovers there is congestion on its path, it reduces the amount of the offered TCP-window that it will fill. For example, if the remote TCP is currently offering a window of 4096-octets, and the local TCP receives a source quench, it will start to act as if the window size is actually 1024, even though the remote TCP continues to advertise 4096. After some period (Nagle suggests waiting for a certain number of acknowledgements) the local TCP resumes using the offered window size. A variation of this algorithm is used in the 4.3BSD UNIX TCP distribution.

continued on next page

Handling Source Quench *(continued)*

Two schemes

Recently researchers have proposed some new congestion control mechanisms that might replace Nagle's. Raj Jain and Van Jacobson have suggested similar schemes to better manage the window size during congestion. Jain's scheme is called "CUTE" (Congestion control Using Timeouts at the End-to-end layer); Jacobson's is called "Slow Start". The basic idea behind both schemes is that instead of shrinking the window until congestion goes away, one should slowly grow the window until encountering congestion. In other words, when congestion is detected, the window size is shrunk to its minimum size, and then allowed to grow according to a special function designed to avoid growing the window size beyond a point where it will cause congestion again. It is worth noting that both schemes use lost packets, *not* the Source Quench message to indicate that congestion is present.

Rate-based flow control

Nagle, Jain and Jacobson all adjust the window size in response to congestion. A different idea called "SQuID" (Source Quench Introduced Delay) comes from Postel and Prue. They suggest that the IP layer should adapt to congestion, by using rate-based flow control. Under rate-based flow control, protocols introduce packets or small groups of packets at specified intervals. (Readers interested in why rate-based flow control is an attractive mechanism should read the SQuID and NETBLT RFCs). Under SQuID, when a Source Quench is received, the IP level increases the interval between the packets being sent to the destination. One possible advantage of SQuID is that it works at the IP level, so that we don't have to add Source Quench algorithms to every transport protocol (TCP, NETBLT, RDP, UDP, etc.) but just add it once to the IP layer. Readers should note that SQuID is officially considered a "wild idea" and is not recommended for implementation.

Congestion avoidance

Finally, before finishing up, I'd like to mention a new concept in the study of network congestion. Jain et al have just published a technical report which proposes a "congestion avoidance" scheme. They observe that by the time a network becomes congested it is already running very poorly. Their congestion avoidance scheme is designed to keep a network operating more efficiently by keeping it well below the congested state. Unfortunately, the scheme would require a change to the IP headers to support, and is thus unlikely to have an impact on IP networks in the immediate future. But their work suggests we should be thinking about avoiding congestion, not just reacting to its presence.

References

Clark, David D., Lambert, Mark L., Zhang, Lixia. "NETBLT: A Bulk Data Transfer Protocol," RFC 998.

Jain, Raj. "A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks," in IEEE Journal on Selected Areas in Communications. October 1986.

Jain, Raj, Ramakrishnan, K.K., and Chiu, Dah-Ming. "Congestion Avoidance in Computer Networks With a Connectionless Network Layer." Digital Equipment Corporation Technical Report, DEC-TR-506. August 1987.

Kline, Charley. "Supercomputers On the Internet: A Case Study," in Proceedings of SIGCOMM87. Association for Computing Machinery.

McKenzie, Alex. "Some Comments on SQuID." RFC 1018.

Nagle, John. "Congestion Control in IP/TCP Internetworks." RFC 896.

Nagle, John. "On Packet Switches With Infinite Storage." RFC 970.

Postel, Jon. "Internet Control Message Protocol." RFC 792

Prue, W., and Postel, J. "Something a Host Could Do with Source Quench: The Source Quench Introduced Delay (SQuID)." RFC 1016.

[Ed: See also "Improving Your TCP: Look at the Timers", in *ConneXions* Volume 1, No. 3, July 1987]

CRAIG PARTRIDGE received his B.A. from Harvard University in 1983, and has been a part-time M.Sc. candidate there since 1986. For the past five years he has worked for BBN Laboratories on a variety of networking related projects including CSNET, the NSF Network Service Center (NNSC), and various projects concerned with distributed systems, IP transport protocols, and network management. In addition, he is a member of the Internet End-To-End Task Force, the Internet Engineering Task Force, and the Distributed Systems Architecture Board Task Force on Naming. He currently splits his time at BBN between CSNET, the NNSC, and managing a small TCP/IP networking project.

2nd TCP/IP Interoperability Conference

December 1-4, 1987, Arlington, VA

The conference is rapidly approaching and the program is now finalized. We'd like to mention a couple of key points in addtion to the information found in the Advance Program:

Network Management meeting

On December 1 there will be a joint meeting of the Network Management and Gateway Monitoring Working Groups of the Internet Engineering Task Force (IETF) from 10:00am to 4:30pm. Anyone interested in internetwork management on TCP/IP networks is invited to attend at no cost. At the meeting members of the working groups will give presentations on the current state of the network management activities. If feasible, there will also be demonstrations of some management systems in action. Copies of the most up-to-date specifications will be available. The chairs of the two working groups, Lee LaBarre of Mitre and Craig Partridge of BBN Labs encourage interested parties to participate.

BOFs

Birds of a Feather (BOF) sessions already scheduled include: *NCSA Telnet for PCs and Macs,* led by Tim Krauskopf from the National Center for Supercomputer Applications; *MVS Performance and Interoperability Issues*, led by Mark Needleman from the Division of Library Automation (DLA) at UC Berkeley; *ISO Development Environment (ISODE) Status and Plans*, led by Marshall Rose from Northrop Research and Technology Center; and *PCs and Macs -- Integrating them into the Internet*, led by John Romkey. If you are interested in conducting a BOF please let us know soon before the slots fill up. For more information call us at 408-996-2042.

CONNEXIONS
21370 Vai Avenue
Cupertino, CA 95014

FIRST CLASS MAIL
U.S. POSTAGE
PAID
CUPERTINO, CA
PERMIT NO. 782